(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

Performance Analysis of Aloha Networks
With Power Capture and Near/Far Effect

by

Joseph T. McCartin

June 1989

Thesis Advisor: Tri T. Ha

Unclassified
Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified		1b Restrictive Markings		
2a Security Classification Authority		3 Distribution Availability of Report Approved for public release; distribution is unlimited.		
2b Declassification/Downgrading Schedule				
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)		
6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol *(If Applicable)* 32	7a Name of Monitoring Organization Naval Postgraduate School		
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000		
8a Name of Funding/Sponsoring Organization	8b Office Symbol *(If Applicable)*	9 Procurement Instrument Identification Number		
8c Address *(city, state, and ZIP code)*		10 Source of Funding Numbers		
		Program Element Number	Project No	Task No / Work Unit Accession No
11 Title *(Include Security Classification)* Performance Analysis of Aloha Networks with Power Capture and Near/Far Effect				
12 Personal Author(s) McCartin, Joseph T.				
13a Type of Report Master's Thesis	13b Time Covered From To	14 Date of Report *(year, month,day)* June 1989		15 Page Count 68
16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	Mobile Data Systems, Aloha, Power Capture, Near/Far Effect

19 Abstract *(continue on reverse if necessary and identify by block number*

This thesis presents an analysis of the throughput characteristics for several classes of Aloha packet networks. Specifically, the throughput for variable packet length Aloha utilizing multiple power levels to induce receiver capture is derived. The results are extended to an analysis of a selective-repeat ARQ Aloha network. Analytical results are presented which indicate a significant increase in throughput for a variable packet network implementing a random two power level capture scheme. Further research into the area of the near/far effect on Aloha networks is included. Improvements in throughput for mobile radio Aloha networks which are subject to the near/far effect are presented. Tactical Command, Control, and Communications (C3) systems of the future will rely heavily on Aloha ground mobile data networks. The incorporation of power capture and the near/far effect into future tactical networks will result in improved system analysis, design, and performance.

20 Distribution/Availability of Abstract [✓ unclassified/unlimited] same as report DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Tri T. Ha	22b Telephone *(Include Area code)* (408) 646-2788	22c Office Symbol 62Ha

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted security classification of this page
All other editions are obsolete Unclassified

Performance Analysis of Aloha Networks With Power Capture and Near/Far Effects

by

Joseph T. McCartin
Captain, United States Air Force
B.S., Worcester Polytechnic Institute, 1984

Submitted in partial fulfillment of the requirements for
the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
(Command, Control, and Communications)

from the

NAVAL POSTGRADUATE SCHOOL
June 1989

Author: Joseph T. McCartin
Joseph T. McCartin

Approved by: Tri T. Ha
Tri T. Ha, Thesis Advisor

Glen A. Myers
Glen A. Myers, Second Reader

[illegible]
Carl Jones, Chairman, Command, Control, and Communications Academic Group

Harrison Shull
Harrison Shull, Provost and Academic Dean

ABSTRACT

This thesis presents an analysis of the throughput characteristics for several classes of Aloha packet networks. Specifically, the throughput for variable packet length Aloha utilizing multiple power levels to induce receiver capture is derived. The results are extended to an analysis of a selective-repeat Automatic Repeat Request (ARQ) Aloha network. Analytical results are presented which indicate a significant increase in throughput for a variable packet network implementing a random two power level capture scheme. Further research into the area of the near/far effect on Aloha networks is included. Improvements in throughput for mobile radio Aloha networks which are subject to the near/far effect are presented. Tactical Command, Control, and Communications (C3) systems of the future will rely heavily on Aloha ground mobile data networks. The incorporation of power capture and the near/far effect into future tactical networks will result in improved system analysis, design, and performance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

United States Department of Defense (DOD) Tactical Command, Control, and Communications (C3) systems of the future will rely heavily on ground mobile data networks. These networks must provide high reliability and connectivity in an environment in which enemy counter-C3 assets will attempt to disrupt and deny friendly C3 networks. A great deal of interest has been focused on Aloha network architectures for future mobile tactical data systems due to their simplicity and robustness.

Research on Aloha has been conducted since the early 1970s [Ref. 1:p. 253]. Most of this work, however has utilized idealized channel models. While this has produced a wealth of information about the general characteristics of Aloha networks, it remains critical that real world conditions be taken into account. Factors which affect Aloha channels in the real tactical environment include fading, Near/Far effect, power capture (both natural and induced), and jamming. This thesis presents analysis of Near/Far effect and power capture on channels which employ the Aloha network discipline.

Chapters II and III concentrate on the effects of induced power capture when employed as part of a variable packet Aloha network. Chapter IV presents an analysis of the ramifications of Near/Far effect on three different Aloha configurations. This work is geared specifically toward ground mobile data network applications.

II. POWER CAPTURE ALOHA

Aloha packet networks have been utilized for some time in data communications systems which are composed of many uncoordinated users, each with low data rate requirements. Recent emphasis on military and commercial ground mobile data networks will increase the number of systems which employ variations of the Aloha protocol as a channel discipline. Although many Aloha systems have been effectively employed to date, several variations on the basic Aloha scheme still remain unstudied. The preponderance of work on Aloha network analysis has focused on an idealized theoretical radio channel, thereby neglecting the effect of real world channel conditions on system performance. It is particularly important to incorporate channel characteristics into Aloha analysis because, in contradiction to intuition, *the overall throughput of an Aloha channel can actually increase under certain contrived or natural channel conditions.* A specific example of this is the case of power capture Aloha.

Another area of neglect in terms of Aloha network analysis is that of variable packet Aloha. Although model results have shown that fixed packet and slotted Aloha both exhibit better throughput characteristics than variable packet, the reality of bit stuffing logic in fact reduces the actual throughput of these disciplines. In contrast, variable packet Aloha can be shown to be a simple and efficient network discipline.

This chapter introduces and develops the throughput characteristics of a variable packet Aloha network which incorporates the power capture phenomenon. For this analysis, only created capture is discussed, although the results can easily be adapted to accommodate natural capture due to fading.

Created capture effects occur when groups of users are assigned different transmitter power levels in an effort to create priority classes, or when all users dynamically select a random transmit power in order to increase channel throughput. Previous analysis of power capture in slotted and fixed packet networks has shown a dramatic increase in channel throughput over the idealized channel models [Ref. 2]. This chapter investigates the effect of created power capture with two random power levels on an Aloha network using variable length packets.

A. SYSTEM DYNAMICS

In this analysis, it is assumed that there is an infinite user population and that the channel traffic rate is Poisson with parameter g packets per second. Assume newly created packets are of variable length x (bits) and are independently and identically distributed according to the probability density function $a(x)$ with mean $\bar{x}$. Within this model, all channel packets are indistinguishable. That is to say that newly arrived packets and retransmitted packets have their length x redrawn afresh from $a(x)$ for every packet. This is an analytical device known as the independence assumption.

The probability that n packets interfere (overlap) with an arbitrary tagged packet during its transmission is given by

$$Pr\{n\} = \frac{[g(x+u)]^n}{n!} e^{-g(x+u)} \qquad (2.1)$$

where x is the tagged packet length, u represents the random length of a preceding packet, and n is the number of interfering packets [Ref. 3]. For the case of $n = 0$, this can be interpreted as the probability that the tagged packet encounters no overlap from preceding packets and the next interarrival is larger than x. Given the case where no interferers can be tolerated by a tagged packet, equation (2.1) can be viewed as the probability of successful tagged packet transmission.

This analysis, however, considers the case when a receiver can be captured by a tagged packet in the presence of interfering packets if the tagged packet signal power exceeds the joint signal power of all interfering packets by some specific capture threshold, γ_0. If at any time during the tagged packet interval, the ratio of tagged packet power to joint interfering power falls below γ_0, the tagged packet is considered to be destroyed.

Unlike slotted Aloha where the *number of interferers is constant* over the entire tagged packet transmission interval (Figure 2.1(a)), or fixed packet Aloha where the number of interferers is a stochastic process made up of early and late interfering packets (Figure 2.1(b)), the number of interferers present in a variable packet Aloha channel represents a stochastic process which is driven by four classes of interferers. Figure 2.1(c) depicts the four possible interferers types. The terms early and late refer to the begin and end time relationship between an interferer and the tagged packet. For instance, the interferer identified as Early-Early in Figure 2.1(c) corresponds to any interferer which begins before the tagged packet (early) and ends before the tagged packet (early).

The dynamics of the interfering stochastic process is as follows. We note that the arrival and departure of n interfering packets partitions the tagged packet interval into several non- overlapping intervals of random length. The exact number of these intervals, given a known number of interferers and their corresponding interferer type, is found by the following relationship.

```
Total Intervals =   (number of Late-Late packets)
                  + (number of Early-Early packets)
```

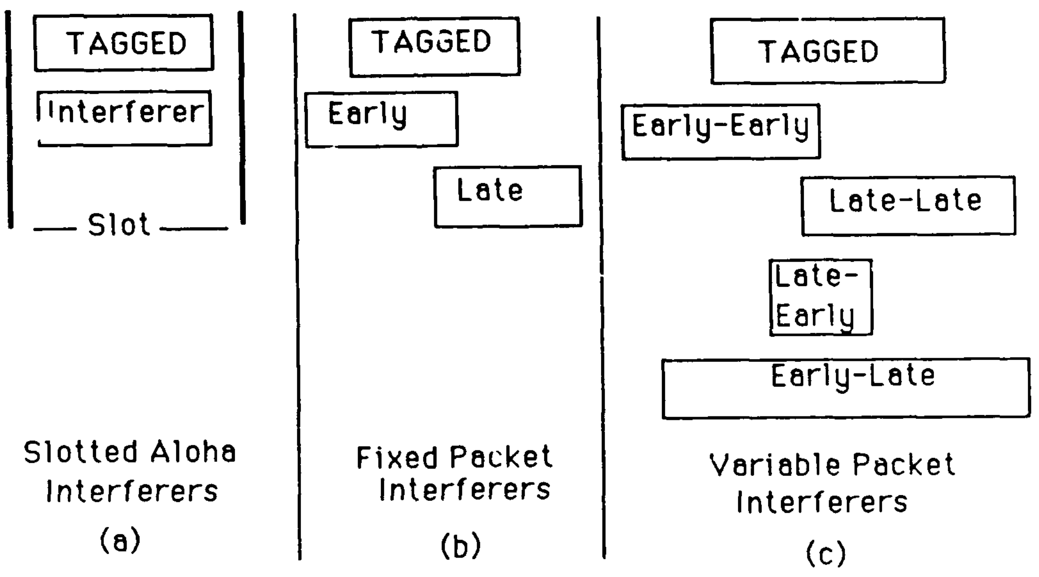


Figure 2.1: Aloha Interfering Packets

```
+ 2 x (number of Late-Early packets) + 1
```

Relating this equation to Figure 2.1(c), it is evident, for instance, that a single Early-Early or Late-Late packet would partition the tagged packet into 1 + 1 intervals. One interval occurs during the absence of the interferer; the other during the presence of an interferer. In a different case, a single Late- Early would partition the tagged packet into 2 + 1 intervals. Two of the partitions would have no interferers present; the other partition would occur during the period of the interfering packet. The Early-Late packet covers the entire duration of the tagged packet and therefore does not provide any partitioning.

Each interval has associated with it a total number of interferers $I(t_i)$, where t_i is the start time of the i'th partition during the tagged packet interval. Each unique sequence of total interferers per partition ($I(t_i)$ for all i) shall be called a realization. Figure 2.2 depicts a specific realization of interfering packets for a case where $n = 3$.

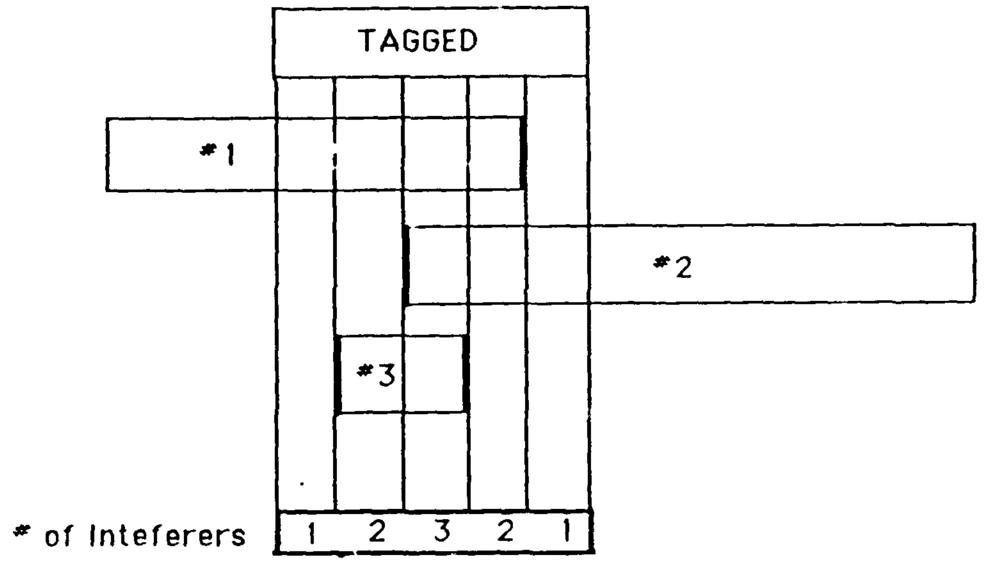


Figure 2.2: An Example of an Interference Realization

For this specific realization, the tagged packet total intervals is 5. The total number ($C(t_i)$) of interferers present during each interval is indicated at the bottom of the figure.

B. MAXIMUM NUMBER OF INTERFERERS

A computer program (Appendix A) was written to generate a list of every possible interfering realization for a given n. From this list of all possible realizations, the number of realizations (C_j) in which the maximum number of interferers present at any time during the tagged packet interval does not exceed some number j was tallied. Table 2.1 gives the results for $n = 1, 2, 3, 4, 5$.

An example of how one would generate a $C_j(n)$ value is as follows. Consider the simple case of $n = 1$. In this case, only four realizations are possible, namely the interfering packet can be an Early-Early, Early-Late, Late-Early, or Late-Late. Continuing with this simple example, it can be seen that given there is 1 known

n	$C_1(n)$	$C_2(n)$	$C_3(n)$	$C_4(n)$	$C_5(n)$	I_T	$f(n)$
1	4	0	0	0	0	4	0.25
2	4	10	0	0	0	14	0.0714
3	4	24	20	0	0	48	0.0208
4	4	50	75	35	0	164	0.0061
5	4	100	225	176	56	561	0.0018

TABLE 2.1: Maximum Number of Interfering Packets Given n

interferer, there is no realization where the maximum number of simultaneous interferers during the entire duration of the realization is less than or equal to $j = 0$. In fact, in this example, the maximum number of simultaneous interferers for each of the four possible realizations is 1. Therefore, the value of $C_1(n)$ for $n = 1$ is 4 as seen in Table 2.1. For the case of $n = 2$, there are 14 combinations of 4 packet types taken 2 at a time. These 14 pairs of interfering packets generate 14 unique realizations. Of these 14 realizations, 4 have a maximum simultaneous number of interferers of 1, and 10 have a maximum of 2 interferers. The program in Appendix A is used to automate the process of generating combinations of packets, constructing the resulting realizations, and determining the maximum number of interferers per realization. Table 2.1 is the result of the program.

Each $C_j(n)$ gives the number of realizations in which the maximum number of interferers at any time equals j, given the total number of interfering packets is known to be n. The function I_T is simply the total number of all possible realizations given n packets are known to interfere. The function $f(n)$ is the inverse of I_T and is simply used as a shorthand notation for $1/I_T$.

C. VARIABLE PACKET ALOHA WITH RANDOM POWER LEVELS

Data packets arrive at the receiver after being spatially attenuated. Each packet is transmitted with one of two normalized power levels given by the set $\Omega = \{1, M\}$. Here, M is some integer multiple of the threshold γ_0 times the lower power level $P = 1$ (normalized), that is $M = N\gamma_0$ where N is an integer greater than 1. Each user selects a transmit power level for each individual packet from the set Ω with equal probability $\frac{1}{2}$ in order to avoid any class prioritization among user groups. The higher power level M is chosen according to the relation

$$(N + 1)\gamma_0 \geq M \geq N\gamma_0 \tag{2.2}$$

where $N \geq 1$ is an integer and γ_0 is the power capture threshold of the receiver.

The tagged packet, arriving at the receiver with power $P_t \in \{1, M\}$ may capture the receiver given a realization of n interfering packets *if and only if*

$$P_t \geq \gamma_0 max[\sum_{j=1}^{I(t_i)} P_j] \tag{2.3}$$

for all intervals (i's) in the realization where P_j is the power level of packet j.

For a tagged packet to capture the receiver, two events must occur. First, the tagged packet must be of power M while none of the interferers are of power M. Call this event A, then

$$Pr\{A|n\} = \frac{1}{2^{(n+1)}} \tag{2.4}$$

The second condition for capture is that the tagged packet must satisfy (2.3) where $P_t = M$ and all $P_j = 1$. Given a specific realization, this equates to

$$N \geq max(I(t_i)) \tag{2.5}$$

The value N is therefore the maximum number of interferers that can be tolerated at any instant of time during the tagged packets transmission interval. If event B denotes the tagged packet power satisfying (2.5), then

$$Pr\{B|n\} = f(n) \sum_{j=0}^{N} C_j(n) \tag{2.6}$$

where $C_j(n)$ is the number of realizations, given n interfering packets, in which the maximum number of simultaneous interferers equals j. Table 2.1 provides a list of $C_j(n)$'s and the inverse $f(n)$ of the total number of possible realizations given n interfering packets.

By using equations (2.4) and (2.6) the conditional capture probability $Pr\{capture|n\}$ is obtained.

$$\begin{aligned} Pr\{capture|n, n = 0\} &= 1 & (2.7) \\ Pr\{capture|n, n \geq 1\} &= Pr\{A|n\}Pr\{B|n\} & (2.8) \\ &= f(n)2^{-(n+1)} \sum_{j=0}^{N} C_j(n) & (2.9) \end{aligned}$$

The overall probability of receiver capture by a tagged packet is found by multiplying the conditional probability in equation (2.9) by the probability that there are n packets in the channel, and summing over all n. Equation (2.1) provides the probability of n packets, therefore, the probability of receiver capture by a tagged packet which shall be called the probability of success is

$$Pr\{success\} = \epsilon^{-g(x+u)}[1 + \sum_{n=1}^{\infty} f(n)2^{-(n+1)}\frac{[g(x+u)]^n}{n!}\sum_{j=0}^{N} C_j(n)] \tag{2.10}$$

The conditional channel departure rate in the interval $x + u$ is given by Reference 3 to be

$$\phi(x,u) = g(x+u)Pr\{success\} \tag{2.11}$$

The channel throughput S, in packets per mean packet time, is related to the channel departure rate by

$$S = \int_0^{\infty}\int_0^{\infty} \frac{1}{2}\phi(x,u)a(x)a(u)dxdu \tag{2.12}$$

Define G to be the attempted packet rate in packets per mean packet time (i.e. $G = g\overline{x}$). Appendix B contains the detailed solution of the above equation. The resulting channel throughput is found to be the following:

$$S = \frac{G}{(G+1)^3} + \sum_{n=1}^{\infty} \frac{n(n+2)}{2^{(n+2)}} f(n)G^{n+1}(G+1)^{-(n+3)}\sum_{j=0}^{N} C_j(n) \tag{2.13}$$

Figure 2.3 shows the throughput of variable packet Aloha with random two power signal for values of $N = 0,1,2,3$. Packet lengths are exponentially distributed and, for computational purposes, n is taken to be between 1 and 5. The curve corresponding to $N = 0$ is for conventional variable packet Aloha.

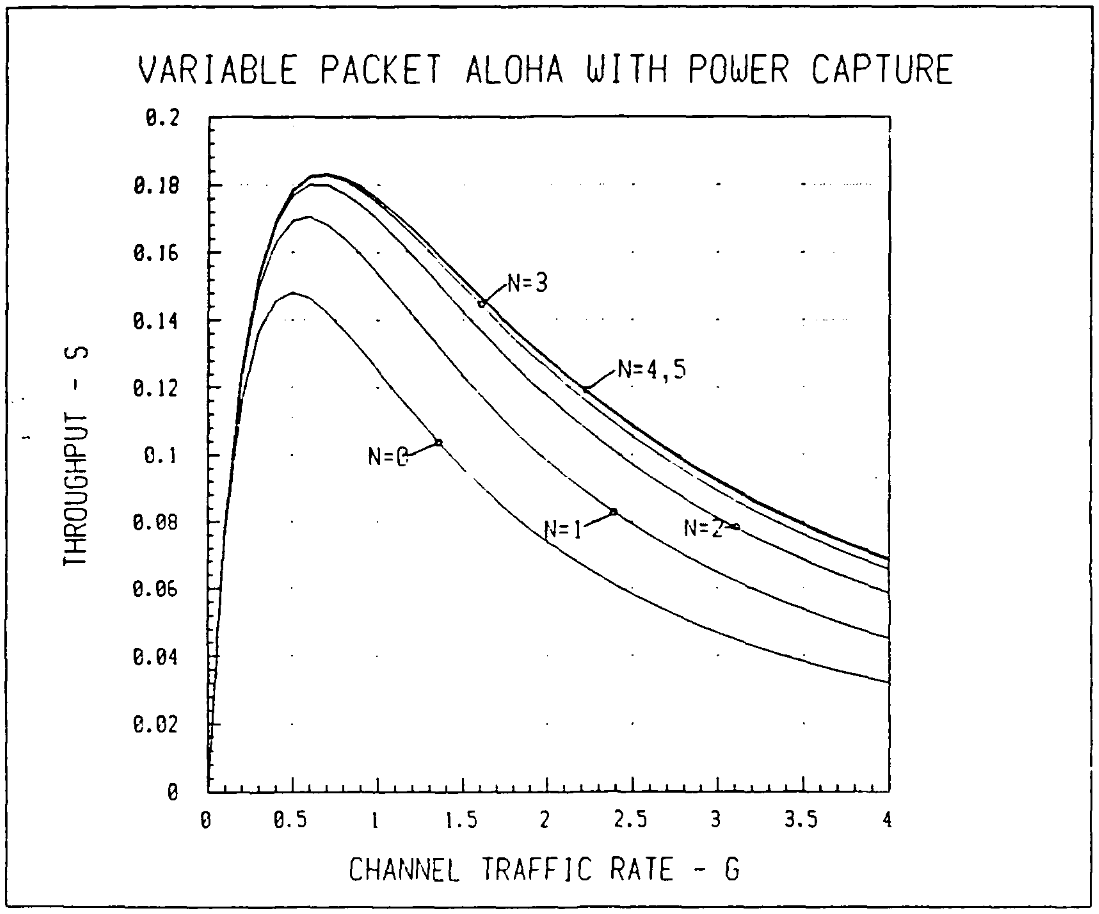


Figure 2.3: Throughput of Variable Packet Aloha With Power Capture

III. SELECTIVE-REPEAT ALOHA WITH CAPTURE

To continue the study of variable packet Aloha, the results of the previous chapter are now applied to a more advanced Aloha network which incorporates selective-repeat as a data link control.

The specific architecture discussed here is a network where remote terminals communicate with a hub station via an Aloha channel in the query-response mode. The feedback hub-to-remote link is an asychronous time-division-multiplexed (ATDM) channel. Each remote terminal receives the same data stream and selects applicable packets by searching for its own address.

Assume that packet length of newly arrived and retransmitted packets in the *Aloha channel* are independent and identically distributed with probability density function $a(x)$ and with mean $\overline{x}$. In this model, packets are indistinguishable. Both newly arrived packets and retransmissions can be pictured as having their length redrawn afresh from $a(x)$ prior to each transmission.

Packets in the ATDM channel are independent and identically distributed with density function $b(y)$ and mean $\overline{y}$.

Variable packets on the Aloha channel are broken up into z numbered and constant length minipackets prior to transmission. When packet damage occurs, only the specific minipackets which are unreadable are retransmitted. Let $D(z), z = 1, 2...$ be the distribution of the minipackets. On the average, each large packet is segmented into $\overline{z}$ minipackets.

Let the channel attempted packet rate be g packets per second and $\bar{z}g$ the channel attempted rate in minipackets per second. With m being the fixed length of a minipacket, the attempted channel rate in minipackets per minipacket time is $m\bar{z}g$.

The conditional probability P_s that a minipacket encountered no collisions is the probability of no overlap from preceding packets and that the next interarrival is larger than the minipacket size m. Thus

$$P_s(z) = e^{-g(mz+m)} \tag{3.1}$$

In a capture environment, however, P_s must take into account the times when the tagged packet captures the receiver in spite of the presence of n interferers. Therefore, using equation (2.1) from the previous chapter, the probability of minipacket success is

$$P_s(z) = e^{-g(mz+m)}[1 + \sum_{n=1}^{\infty} f(n) 2^{-(n+1)} \frac{[g(mz+m)]^n}{n!} \sum_{j=0}^{N} C_j(n)] \tag{3.2}$$

The conditional probability of successful minipacket transmission is

$$q(z,y) = P_s(z)[1 - p_1(m)][1 - p_2(y)] \tag{3.3}$$

where $p_1(m)$ is the minipacket error probability due to errors in the Aloha channel and $p_2(y)$ is the error rate in the ATDM channel. The conditional minipacket departure rate in the interval of $z+1$ minipackets is

$$o_m(z,y) = mg(z+1)q(z,y) \tag{3.4}$$

The average minipacket throughput is given by the following equation:

$$S_m = \frac{\bar{z}}{\bar{z}+1} \int_0^{\infty} o_m(z,y)dy \tag{3.5}$$

Assume that each packet consists of a geometrically distributed number of fixed length minipackets, then the distribution $D(z)$, which is the probability that a packet consists of z minipackets, is defined as follows.

$$D(z) = (1-\mu)^{z-1}\mu \tag{3.6}$$

where μ is the probability that a minipacket is generated. Because $\bar{z} = \frac{1}{(1-\mu)}$, $D(z)$ can be written as

$$D(z) = \frac{1}{\bar{z}}(1 - \frac{1}{\bar{z}})^{n-1} \tag{3.7}$$

Appendix C provides the detailed solution for the throughput equation. The resulting rather lengthy set of equations is as follows:

$$S_m = A + \frac{G}{\bar{z}+1}\sigma(1 - p_1(m)) \sum_{n=1}^{\infty} h(n) w^{-2}[F(n)R(n) - \beta] \tag{3.8}$$

$$A = \frac{\sigma G v^2(\alpha - v)(1 - p_1(m))}{\bar{z}(\bar{z}+1)w^2(1-v)^2} \tag{3.9}$$

$$F(n) = (1-v)^{-(t+1)}(t+1)! \tag{3.10}$$

$$R(n) = \sum_{r=1}^{t} v^r \sum_{k=0}^{r-1} (-1)^k E(k,r) \tag{3.11}$$

$$E(k,r) = \frac{(r-k)^t}{k!(t-k+1)!} \tag{3.12}$$

$$G = m\bar{z}g \tag{3.13}$$

$$h(n) = f(n)2^{-(n+1)}\frac{G^n}{n!\overline{z}^{(n+1)}}\sum_{j=0}^{N} C_j(n) \tag{3.14}$$

$$v = (1 - \frac{1}{\overline{z}})e^{-(\frac{G}{\overline{z}})} \tag{3.15}$$

$$w = (1 - \frac{1}{\overline{z}}) \tag{3.16}$$

$$k = z + 1 \tag{3.17}$$

$$t = n + 1 \tag{3.18}$$

Figure 3.1 shows the throughput for the variable packet Aloha with selective-repeat data link control. The parameters are: $p_1(m) = 0$, $p_2(m) = 0$, x and y are both exponentially distributed, $\overline{x} = \overline{y}$, and $\overline{z} = 8$ minipackets.

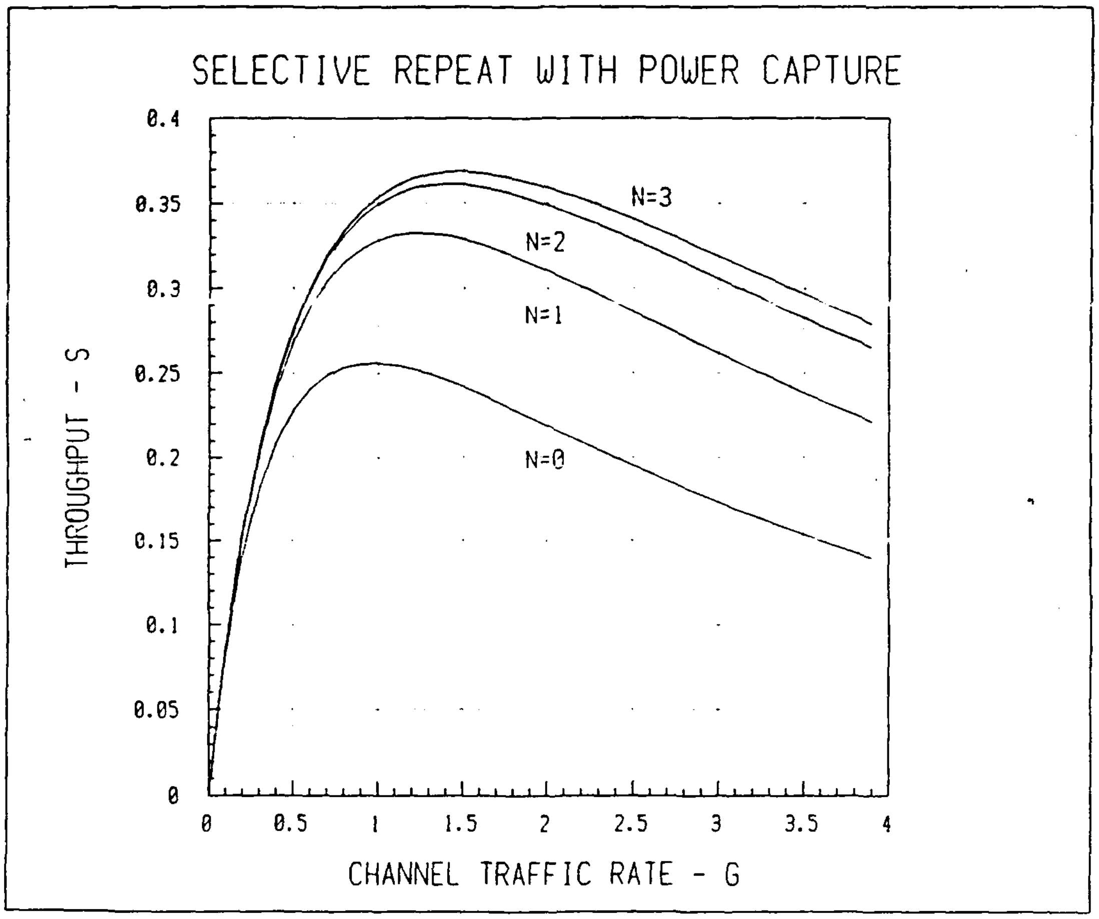


Figure 3.1: Throughput of Selective-Repeat Aloha with Power Capture

IV. NEAR/FAR EFFECT ON ALOHA MODELS

This final analysis chapter deals with a network phenomenon which is specific to mobile packet radio networks that do not incorporate adaptive control of terminal transmit power. *The near/far effect is a condition which occurs when packets arrive at a receiver with different mean power due to differences in transmission distances.* Under these conditions, a tagged packet may be able to capture the receiver, in spite of interfering packets, if the tagged packet power is sufficiently greater than the joint power of the interfering packets. Because each packet is spatially attenuated due to the near/far effect, all packets have a finite probability of capturing the receiver. The throughput characteristics of various Aloha network configurations which are subject to the near/far effect are derived and presented.

A. CELL MODEL

The mean power w of a packet at a distance r from the transmitter is of the general form

$$w = cr^{-\alpha} \tag{4.1}$$

In the event of ground-wave propagation without shadowing,

$$c = P_{T_i} G_{T_i} G_R H_{T_i}^2 H_R^2 \tag{4.2}$$

where P_{T_i}, G_{T_i}, and H_{T_i} are the transmit power, antenna gain, and antenna height (above ground), respectively, of the transmitting mobile terminal. The values G_R and H_R are the gain and height above ground of the base station antenna. The exponent α gives the attenuation law for the channel considered ($2 \leq \alpha \leq 5$). In the event of UHF propagation, a typical value is $\alpha = 4$ [Ref. 3:p. 264].

Assuming identical mobile terminals and omnidirectional antennas c is normalized to unity for all users without loss of generality since receiver capture power is determined by the ratio of signal powers [Ref. 3:p. 264]. Additionally, the value of α in this analysis is taken as 4.

In this model, signal power depends on the spatial distribution of users in the network. Users create a traffic density $g(r)$ within a circular cell around a base station receiving node. The density $g(r)$ can be thought of as the normalized packet traffic per unit area at a distance r from the base receiver. The packet generation rate in a given area depends on the distance to the base station r and is independent of direction. Given this cell model, the total traffic rate offered in the network can be described as

$$G = 2\pi \int_0^\infty x g(x) dx \tag{4.3}$$

Figure 4.1 depicts the general cell model used in this analysis.

The spatial distribution of packet generation is found to be the probability a packet is generated within a distance R. That is to say

$$F_R(r) = Pr\{R \leq r\} = \frac{2\pi}{G} \int_0^r x g(x) dx \tag{4.4}$$

Differentiating, the probability density function for packet generation is found to be

$$f_R(r) = \frac{2\pi}{G} r g(r) \tag{4.5}$$

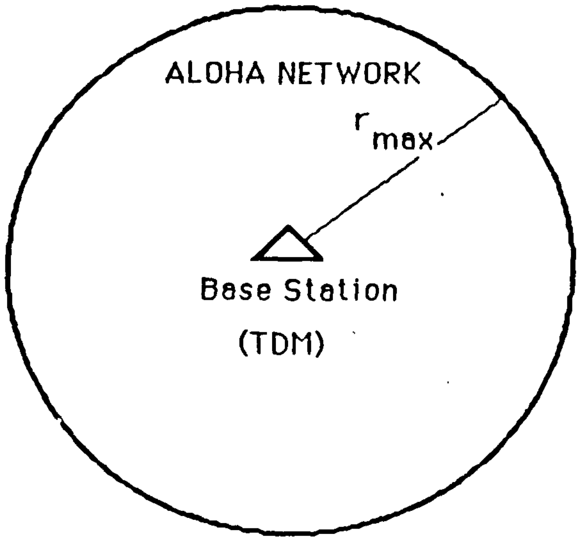


Figure 4.1: Cell Model for Near/Far Effect Analysis

As a means of simplification, users are assumed to be uniformly distributed in a circle of radius r_{max} with the base receiving station at the center of the cell as shown in Figure 4.2. As seen in Figure 4.2, $g(r)$ is normalized to $G/\pi r_{max}^2$ for $0 \leq r \leq r_{max}$.

From this, the equations for the total network traffic and packet generation reduce to

$$G = \pi r_{max}^2 \tag{4.6}$$

$$f_R(r) = \frac{2\pi r}{G} \tag{4.7}$$

for $0 \leq r \leq r_{max}$

A standard change of variable operation is performed on equations (4.1) and (4.6) to define a probability density function for the packet power w.

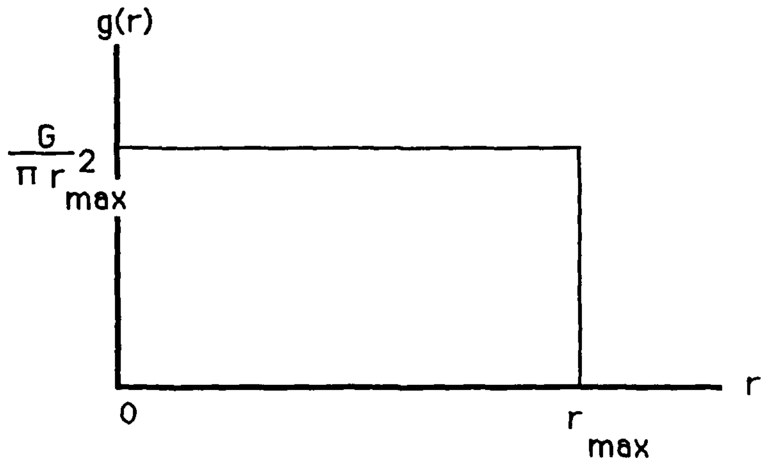


Figure 4.2: Distribution of Users Within the Cell

$$f_W(w) = \frac{2\pi}{\alpha G} w^{-(\frac{2}{\alpha}+1)} \tag{4.8}$$

for $r_{max}^{-\alpha} \leq w \leq \infty$.

At this point, the network cell model has been described with the distribution of packet generation given by equation (4.6) and the packet power density function given by equation (4.7). The near/far effect is now incorporated into the model.

B. NEAR/FAR EFFECT

In a phenomenon which is similar in many respects to the power capture case described earlier, the near/far capture occurs when the ratio of a tagged packets power to the joint interfering power exceeds some receiver capture threshold. Again, the number of interferers during the tagged packets transmission is a stochastic process. The capture ratio is described as

$$Z = \frac{X}{Y} \tag{4.9}$$

where X is the packet power of the tagged packet and Y is the joint interfering power of some number n interfering packets. As in previous discussions, the receiver is captured if the ratio described by Z exceeds the threshold given by the parameter γ_0.

The distribution of X is simply the single packet power density function given in (4.7). The joint interfering power density Y is dependent on the number of interferers and is given as

$$f_Y(y|n) = [f_W(w)]^{n\otimes} \tag{4.10}$$

where $n\otimes$ denotes the n-fold convolution. For the cases of $n = 0$ and $n = 1$, no convolutions are required to determine $f_Y(y)$. For higher numbers of interferers, a computer based convolution package is used to generate specific values for $f_Y(y|n)$. Appendix D provides details on the analytical result for the case of $n = 1$ and the numerical results for $n = 2, 3$.

Utilizing the density functions for X and Y, the distribution of Z can be found using the following formula.

$$f_Z(z|n) = \int_0^\infty y f_X(yz) f_Y(y|n) dy \tag{4.11}$$

The capture probability, that is the probability that Z exceeds the receiver threshold γ_0 given n interferers, is found by integrating $f_Z(z|n)$ from γ_0 to ∞.

Table 4.1 presents the probability of capture given n interferers for the range $0 \leq n \leq 4$. Although the case of $n = 1$ is in a closed form with parameter γ_0, all subsequent capture probabilities assume a value of $\gamma_0 = 3$.

n	$Pr\{capture\|n\}$
0	1
1	$\frac{1}{2}\gamma_0^{-2/\alpha}$
2	0.057
3	0.022
4	0.010

TABLE 4.1: Capture Probabilities

The limited list of capture probabilities listed in Table 4.1 is sufficient to provide excellent resolution of the throughput characteristics for the Aloha models discussed in the following sections. The probability of capture for cases where there are greater than 4 interfering packets is so small as to be considered insignificant.

C. ALOHA NETWORK MODELS

Three common Aloha models are analyzed with respect to the improvement in overall network throughput gained from the near/far effect. In each case, the general definition of throughput is

$$S = GPr\{success\} \tag{4.12}$$

which in a capture environment can be viewed as

$$S = G \sum_{n=0}^{\infty} Pr\{capture|n\} Pr\{n\} \qquad (4.13)$$

The following sections use this approach to determine the network throughput for slotted and fixed packet Aloha networks. Variable packet Aloha is analyzed in the same general manner as was done in Chapter I for power capture.

1. Slotted Aloha

For slotted Aloha, the probability that n packets are generated during a given packet time is given by the Poisson Distribution [Ref. 1:p. 255]:

$$Pr\{n\} = \frac{G^n \epsilon^{-G}}{k!} \qquad (4.14)$$

Therefore, the general form for the throughput in a slotted Aloha network with the near/far effect is

$$S = G \sum_{n=0}^{\infty} Pr\{capture|n\} \frac{G^n \epsilon^{-G}}{n!} \qquad (4.15)$$

Using the $Pr\{capture|n\}$ values in Table 4.1, the throughput S is calculated and is shown in Figure 4.3.

2. Fixed Packet Aloha

The number of packets generated during a given packet time in a fixed packet network is given by [Ref. 4:p. 11]:

$$Pr\{n\} = \frac{(2G)^n}{n!} \epsilon^{-2G} \qquad (4.16)$$

resulting in a throughput of

$$S = G \sum_{n=0}^{\infty} pr\{capture|n\} \frac{(2G)^n}{n!} \epsilon^{-2G} \qquad (4.17)$$

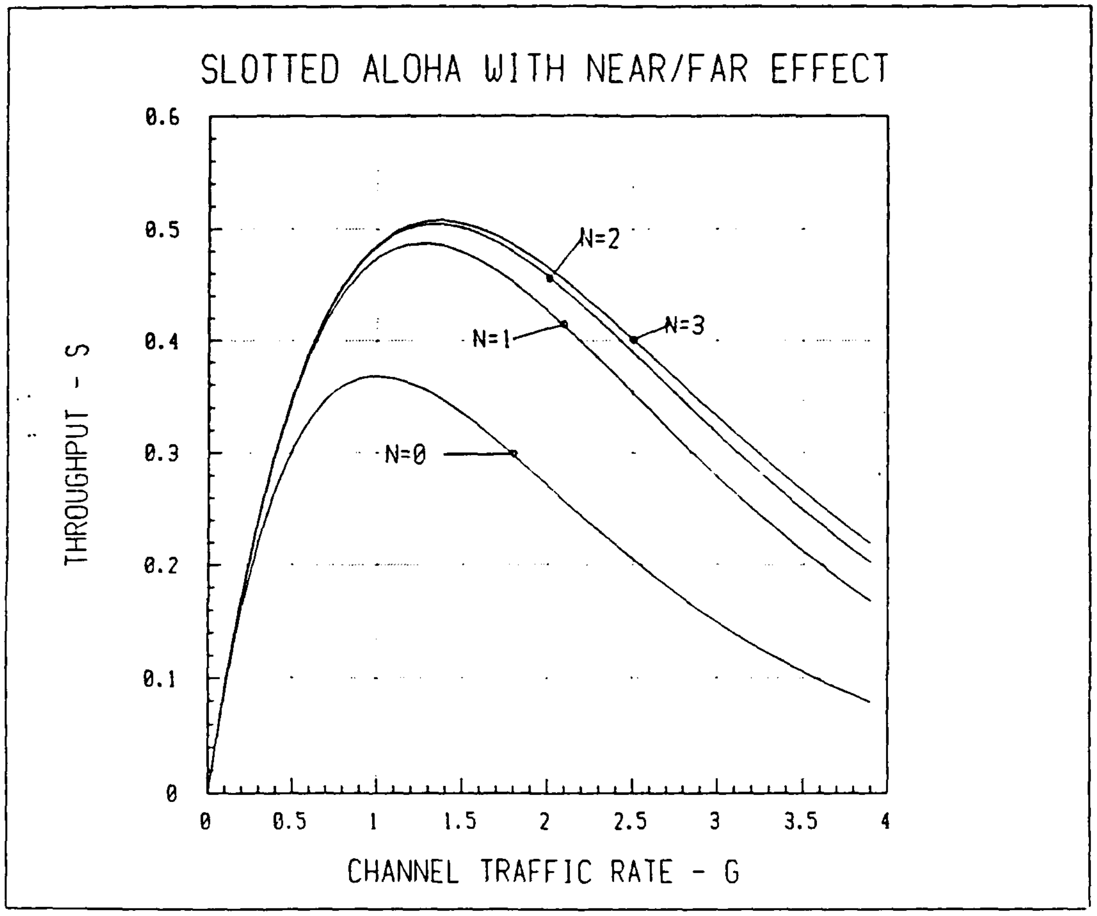


Figure 4.3: Slotted Aloha With Near/Far Effect

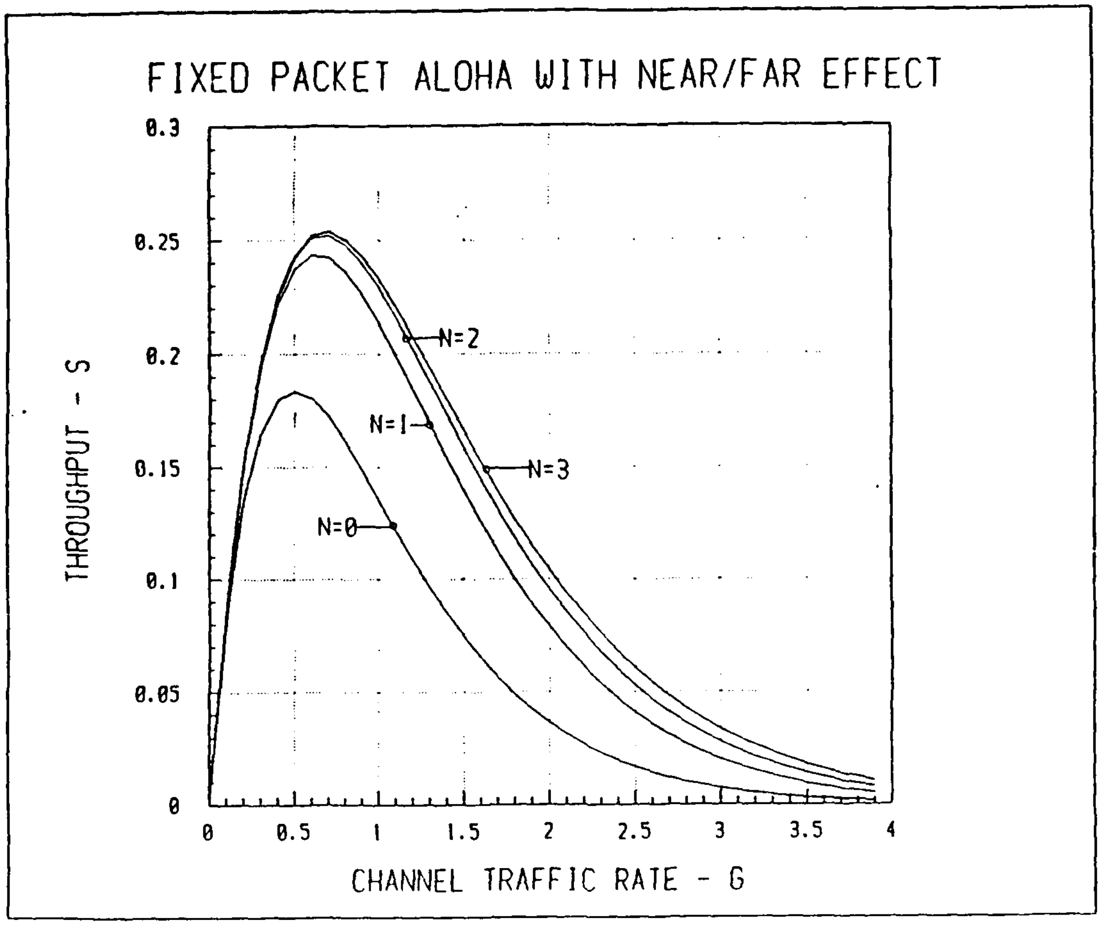


Figure 4.4: Fixed Packet Aloha With Near/Far Effect

Again, using the numerical results for the probability of capture given in Table 4.1, the throughput is calculated and is depicted in Figure 4.4.

3. Variable Packet Aloha

This analysis is very similar to the work presented in Chapter II and therefore the variable names used here are the same. First, the probability of success for a tagged packet is defined as

$$P_s(x,v) = \sum_{n=0}^{\infty} Pr\{capture|n\} e^{-g(x+v)} \frac{[g(x+v)]^n}{n!} \tag{4.18}$$

For this analysis, perfect channel conditions are assumed in order to simplify the calculations. The conditional probability of a successful transmission in this case is seen to be

$$q(x,u) = P_s(x,u) \tag{4.19}$$

The conditional channel departure rate in the interval $x + u$ is

$$\phi(x,u) = g(x+u)q(x+u) \tag{4.20}$$

From this, the channel throughput can be calculated from the following equation:

$$S = \frac{1}{2}\int_0^\infty \int_0^\infty \phi(x,u)a(x)a(u)dxdu \tag{4.21}$$

Take the case where $a(x)$ and $a(u)$ are exponentially distributed, the above equation simplifies to

$$S = \frac{G}{(G+1)^3} + \sum_{n=1}^{\infty} \frac{n(n+2)}{2} Pr\{capture|n\} G^{n+1}(G+1)^{-(n+3)} \tag{4.22}$$

Using the numerical results for $Pr\{capture|n\}$ given in Table 2.1, the throughput for variable packet Aloha with Near/Far effect is shown in Figure 4.5.

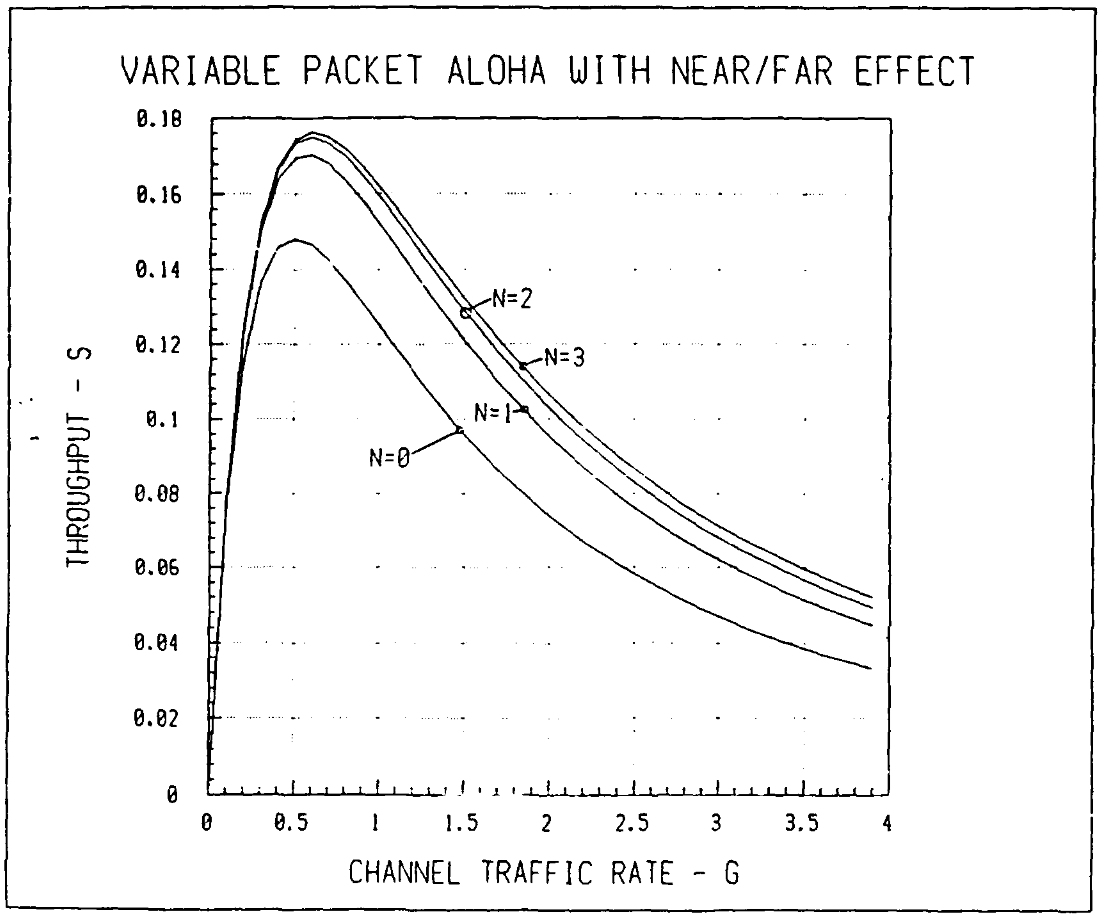


Figure 4.5: Throughput for Variable Packet Aloha With Near/Far Effect

V. CONCLUSIONS

This thesis has demonstrated the effects of induced two level power capture on variable packet Aloha networks and the effects of the near/far phenomenon on various Aloha configurations. In all cases, significant improvements in overall network throughput were achieved.

The development of highly reliable Tactical data systems of the future depend on Aloha as a multiaccess scheme for ground mobile terminals. It is critical that these networks be allowed to achieve their maximum potential. Given the increased reliance on tactical data communications, Aloha networks which provide the maximum user throughput possible, even in a jamming environment, must be developed.

The increases in overall network throughput presented in this thesis indicated the improvements in performance which can be realized when power capture and the near/far effect are incorporated. As seen in Chapter 2, induced power capture increased the network throughput of a variable packet Aloha network from 0.148 packets per mean packet time in the idealized case to 0.181 in a capture environment. Utilizing the selective-repeat data link control with power capture variable packet Aloha increased throughput from a maximum of 0.256 to about 0.354. In the final analysis chapter, improvements in throughput for slotted, fixed packet and variable packet Aloha which incorporated the near/far effect were presented.

Continued work in the area of Aloha network analysis will be needed in order to develop the high capacity, high reliability tactical data networks required in the future. Incorporating power capture and the near/far effect into future designs will result in better systems analysis and design for these tactical mobile networks.

APPENDIX A

```
{*******************************************************}
{                                                       }
{  PROGRAM CJ                                           }
{                                                       }
{  Joe McCartin, January 1989                           }
{                                                       }
{  Turbo Pascal - IBM PC                                }
{                                                       }
{  Description:                                         }
{                                                       }
{    This program is designed to generate all           }
{  realizations of interferers given n packets are      }
{  known to interfere. Using the realizations, a        }
{  table of Cj(n) values is compiled.  This program     }
{  is geared specifically toward the purposes of this   }
{  thesis and is not a general purpose program.         }
{                                                       }
{*******************************************************}

program cj (filevar, output);
   const
      max_intervals = 11:
      matrix_length = 1024;
      cj_matrix_dim = 10;

   type
      realization_vector = array [1..max_intervals] of
                                                integer;
      realization_matrix = array [1..matrix_length] of
                           realization_vector;
      cj_vector = array [1..cj_matrix_dim] of integer;
      cj_array = array [1..cj_matrix_dim] of cj_vector;
      index = array [1..4] of integer;
      comb_matrix = array [1..220] of index;

   var
      realization_x,
      realization_y:realization_matrix;

      vector:comb_matrix;
      vector_length:integer;

      active_matrix,
      active_matrix_length:integer;

      cj_matrix:cj_array;
      init_interferers,
      max_interferers,
      min_interferers,
      num_intervals,
      final_interferers:integer;

      n,
      i,
      j,
      t,
      h,
      runs:integer;

      file_name:string[10];
      filevar:text;
      outfilevar:text;

      ans:char;

      archive:boolean;

{*******************************************************}
{                                                       }
{  INITIALIZE_CJ_MATRIX                                 }
{                                                       }
{  Tabulated values of Cj's are kept in a square matrix }
{  defined by the dimension cj_matrix_dim.  This proc   }
{  initialize this matrix which is called CJ_MATRIX.    }
{                                                       }
{*******************************************************}

procedure INITIALIZE_CJ_MATRIX;

   var
      i,
      j:integer;

   begin
      for i:= 1 to cj_matrix_dim do
         begin
            for j:= 1 to cj_matrix_dim do
               begin
                  cj_matrix [i,j] := 0;
               end;
         end;
   end;
{******************************************************}
{                                                      }
{  INIT_REALIZATION_MATRIX                             }
{                                                      }
{  In this program, realizations are generated one     }
{  interval at a time until the entire realization is  }
{  constructed. For each interval, a matrix of all     }
{  known realizations is read. Additions to these      }
{  partial realizations are then placed in a different }
{  matrix. This proc initialize both of these         }
{  matrices.                                           }
{                                                      }
{******************************************************}

procedure INIT_REALIZATION_MATRIX;
   var
      i,
      j:integer;

   begin
      writeln ('INITIALIZING REALIZATION MATRICES ');
      writeln;
      for i := 1 to matrix_length do
         begin
            for j := 1 to max_intervals do
               begin
                  realization_x[i,j] := 0;
                  realization_y[i,j] := 0;
               end;
         end;

      writeln ('INITIALIZATION COMPLETE');
      writeln;
   end;


{******************************************************}
{                                                      }
{  COPY_X_TO_Y                                         }
{                                                      }
{  This procedure copies the contents of realization   }
{  matrix X into realization matrix Y.                 }
{                                                      }
{******************************************************}

procedure COPY_X_TO_Y ( x,y:integer);
   var
      i:integer;

   begin
      for i := 1 to max_intervals do
         realization_y[y,i] := realization_x[x,i];
   end;


{******************************************************}
```

```
{                                                     }
{  COPY_Y_TO_X                                        }
{                                                     }
{  This procedure copies the contents of realization  }
{  matrix Y into realization matrix X                 }
{                                                     }
{*****************************************************}

procedure COPY_Y_TO_X (y,x:integer);
   var
      i:integer;

   begin
      for i := 1 to max_intervals do
         realization_x[x,i] := realization_y[y,i];
   end;


{****************************************************}
{                                                    }
{  PRINT_MATRIX                                      }
{                                                    }
{  Print_matrix prints out the most recently updated }
{  realization matrix (X or Y).  If the flag MATRIX is }
{  set to 1, then the REALIZATION_X was the last     }
{  updated and is subsequently printed.  If the flag }
{  is not 1, the Y matrix is printed.  If the        }
{  diagnostic flag ARCHIVE is true, output goes both }
{  to the screen and to the data file OUTFILEVAR.    }
{  Else, output is to screen only.                   }
{                                                    }
{****************************************************}

procedure PRINT_MATRIX (matrix, length, width:integer);

   var
      i,
      j:integer;

   begin
      for i := 1 to length do
         begin
            writeln;
            if archive then
               writeln (outfilevar);
            for j := 1 to width do
               begin
                  if matrix = 1 then
                     begin
                        write (realization_x[i,j]);
                        if archive then
```

```
                              write (outfilevar,
                                     realization_x[i,j]);
                        end
                     else
                        begin
                           write (realization_y[i,j]);
                           if archive then
                              write (outfilevar,
                                     realization_y[i,j]);
                        end;
                  end;
            end;
         writeln;
         if archive then
            writeln (outfilevar);
      end;


   {*******************************************************}
   {                                                       }
   {  GENERATE_REALIZATIONS                                }
   {                                                       }
   {  This procedure produces a matrix containing all      }
   {  possible realizations given an initial number of     }
   {  interferers, a maximum number of interferers, and    }
   {  the total number of intervals.  A tree method is     }
   {  employed to build up a realization one interval at   }
   {  a time.  The first interval contains the initial     }
   {  number of interferers.  This partial realization     }
   {  is placed in a realization matrix.  On the next      }
   {  pass, the partial realization is read and the next   }
   {  interval is added to it.  These realizations are     }
   {  then placed into another realization matrix.  This   }
   {  process continues until all intervals have been      }
   {  filled in for all realizations.                      }
   {                                                       }
   {*******************************************************}

   procedure GENERATE_REALIZATIONS (init_interferers,
                    max_interferers,
                                   num_intervals,
                 min_interferers:integer;
                                   var active_matrix:integer;
                                   var active_matrix_length
                                :integer);

      var
         temp,
         i,
         j,
         k,
      x_list_length,
      y_list_length,
      interval_count:integer;

   begin

      writeln ('GENERATING REALIZATIONS FOR ',max_interferers,'
            MAX INTERFERERS AND ',num_intervals,' INTERVALS');
{      writeln (init_interferers,' ',max_interferers,'
   ',num_intervals,' ',min_interferers);}
      writeln;

      if archive then
         begin
            writeln (outfilevar);
            writeln (outfilevar,
                     'REALIZATIONS **************');
         end;

      realization_x[1,1] := init_interferers;
      x_list_length := 1;
      interval_count := 1;
      active_matrix := 1;
      active_matrix_length := 1;

      while interval_count < num_intervals do
         begin
            y_list_length := 0;
{            writeln ('within x to y loop');}
            for i := 1 to x_list_length do
               begin
                  if realization_x[i,interval_count] <
                       max_interferers then
                     begin
                        y_list_length := y_list_length + 1;
                        copy_x_to_y (i,y_list_length);

realization_y[y_list_length,interval_count+1]
                   :=  realization_x[i,interval_count]+1;
                     end;

                  if realization_x[i,interval_count] >
                       min_interferers then
                     begin
                        y_list_length := y_list_length + 1;
                        copy_x_to_y(i,y_list_length);

realization_y[y_list_length,interval_count+1]
                   :=  realization_x[i,interval_count]-1;
                     end;
                  end;
               interval_count := interval_count + 1;
               active_matrix := 2;
               active_matrix_length := y_list_length;

               x_list_length := 0;

               if interval_count < num_intervals then
                  begin
                     for k := 1 to y_list_length do
                        begin
{                            writeln (' within y to x loop');}
                           if realization_y[k,interval_count]
                                         < max_interferers then
                              begin
                                 x_list_length := x_list_length
                                                + 1;
                                 copy_y_to_x(k,x_list_length);

realization_x[x_list_length,interval_count+1]
                     :=  realization_y[k,interval_count]+1;
                              end;
                           if realization_y[k,interval_count]
                                       > min_interferers then
                              begin
                                 x_list_length := x_list_length
                                                + 1;
                                 copy_y_to_x(k,x_list_length);

realization_x[x_list_length,interval_count+1]
                  :=  realization_y[k,interval_count]-1;
                              end;
                         end;
                     interval_count := interval_count + 1;
                     active_matrix := 1;
                     active_matrix_length := x_list_length;
                  end;

             end;
             writeln ('There Were ',active_matrix_length,'
            Realizations');
             if archive then
                begin
                   writeln (outfilevar,'There Were
',active_matrix_length,' Realizations');            end;
             if active_matrix = 1 then
                print_matrix (1,active_matrix_length,
                              interval_count)
             else
                print_matrix (2,active_matrix_length,
                              interval_count);
```

```
    end;

{*****************************************************}
{                                                     }
{  FILTER                                             }
{                                                     }
{  This procedure accepts a realization matrix which  }
{  contains all possible realizations given a specific }
{  combination of interferers.  Each realization in   }
{  the matrix is checked for the invalid condition of }
{  overcount.  An overcount occurs when a realization }
{  has indicated that a certain number of interferers }
{  has departed but yet somewhere in the realization  }
{  the interferer count goes beyond the count of the  }
{  remaining interferers.  Realizations which are     }
{  filtered out are discarded.  All good realizations }
{  are placed into the currently inactive realization }
{  matrix.                                            }
{                                                     }
{*****************************************************}

procedure FILTER (var length:integer; intervals,
interferers:integer;
                  var active_matrix:integer);

var
   i,
   j,
   p,
   max,
   num_final_interferers:integer;

   invalid_realization:boolean;

begin
   writeln;
   writeln ('FILTERING THE REALIZATION MATRIX ');
   writeln;


   j := 1;

   if active_matrix = 1 then
      begin
         active_matrix := 2;
         for i := 1 to length do
            begin
               invalid_realization := false;
               num_final_interferers :=
                                realization_x[i,intervals];
                  if num_final_interferers = interferers then
                     begin
                        if intervals > 2 then
                           begin
                              max := max_interferers;
                              for p := 1 to (intervals - 1) do
                                 begin
                                    if realization_x[i,p] >
                                       realization_x[i,p+1] then
                                       max := max - 1;
                                    if realization_x[i,p+1] > max
                                    then
                                       invalid_realization
                                               := true;
                                 end;
                           end;
                        if invalid_realization = false then
                           begin
                              copy_x_to_y (i,j);
      {                        print_matrix(2,j);}
                              j := j + 1;
                           end;
                     end;
                 end;
      end
   else
      begin
         active_matrix := 1;
         for i := 1 to length do
            begin
               invalid_realization := false;
               num_final_interferers :=
                             realization_y[i,intervals];
               if num_final_interferers = interferers then
                  begin
                     if intervals > 2 then
                        begin
                           max := max_interferers;
                           for p := 1 to (intervals - 1) do
                              begin
                                 if realization_y[i,p] >
                                   realization_y[i,p+1] then
                                    max := max - 1;
                                 if realization_y[i,p+1] > max
                                 then
                                    invalid_realization
                                             := true;
                              end;
                        end;
                     if invalid_realization = false then
                        begin
```

```
                              copy_y_to_x (i,j);
      {                        print_matrix (1,j);}
                              j := j + 1;
                            end;
                        end;
                    end;
                end;
            writeln;
            writeln ('THERE WERE ',j-1,' VALID REALIZATIONS FOUND');
            writeln;
            if archive then
               begin
                  writeln (outfilevar);
                  writeln (outfilevar,'FILTER ********************');
                  writeln (outfilevar,'There Were ',j-1,' Valid
                Realizations');
               end;
            length := j-1;
            print_matrix (active_matrix,length,intervals);
            writeln;
            if archive then
               writeln (outfilevar);
         end;


{*****************************************************}
{                                                     }
{  GET_Cj                                             }
{                                                     }
{  GET_Cj reads in all valid realizations individually }
{  and determines the maximum number of interferers   }
{  in each realization.  A tally is kept of how many  }
{  realizations have a given maximum number of        }
{  interferers.                                       }
{                                                     }
{*****************************************************}

procedure get_cj (active_matrix, active_matrix_length,
                  n: integer);

   var
      i,
      j,
      max:integer;

   begin
      writeln ('FINDING Cjs FOR A REALIZATION WITH ',n,'
       PACKETS');

      if archive then
         begin
```

```
          writeln (outfilevar);
          writeln (outfilevar,'CJ CALCULATION *********');
          writeln (outfilevar);
       end;

    for i := 1 to active_matrix_length do
       begin
          if active_matrix = 1 then
             begin
                max := realization_x[i,1];
                for j := 2 to max_intervals do
                   begin
                      if  max < realization_x[i,j] then
                         max := realization_x[i,j];
                   end;
                cj_matrix [n,max] := cj_matrix [n,max] + 1;
                if archive then
                   begin
                      writeln (outfilevar);
                      writeln (outfilevar, 'MAX FOR THIS PASS
                          IS ',max);
                      writeln (outfilevar,'CUMULATIVE MAX FOR
                          I=',max,' IS ',cj_matrix[n,max]);
                   end;
             end
          else
             begin
                max := realization_y[i,1];
                for j:= 2 to max_intervals do
                   begin
                      if max < realization_y[i,j] then
                         max := realization_y[i,j];
                   end;
                cj_matrix[n,max] := cj_matrix[n,max] + 1;
                if archive then
                   begin
                      writeln (outfilevar);
                      writeln (outfilevar, 'MAX FOR THIS PASS
                          IS ',max);
                      writeln (outfilevar,'CUMULATIVE MAX FOR
                          I=',max,' PKTS=',n,' IS
             ',cj_matrix[n,max]);
                   end;
             end;
       end;
    end;


{*******************************************************}
{                                                       }
{  PRINT_CJ                                             }
{                                                      }
{  This procedure prints out the matrix of Cj values   }
{  constructed by the procedure GET_Cj.                }
{                                                      }
{******************************************************}

procedure PRINT_CJ;

   var
      i,
      j:integer;

   begin
      writeln ('    |  1  2  3  4  5  6  7  8  9  10');
      for i:= 1 to cj_matrix_dim do
         begin
            write (i,'   |');
            if archive then
               write (outfilevar,i,'   |');
            for j:= 1 to cj_matrix_dim do
               begin
                  write(' ',cj_matrix[i,j],' ');
                  if archive then
                     write (outfilevar,' ',cj_matrix[i,j]);
               end;
            writeln;
            if archive then
               writeln (outfilevar);
         end;
   end;


{******************************************************}
{                                                      }
{  COMB                                                }
{                                                      }
{  Given a total number of interferers, every          }
{  combination of interferer packet type must be       }
{  determined.  This procedure accepts the total       }
{  number of interferers n and generates every         }
{  combination of the four interferer types which      }
{  add up to the total n.  This is basically an        }
{  enumeration of the number of combinations of 4      }
{  things taken n at a time.  This algorithm is        }
{  derived from Reference 7.                           }
{                                                      }
{******************************************************}

procedure COMB (n,k:integer; var r:index; var mtc:boolean);
   label  10, 11, 15, 20, 30, 31;
   var
      i,
      j,
      w,
      x,
      y,
      z,
      l:integer;

   begin

10:   if mtc = true then
         begin
            goto 20;
         end
      else
         begin
            r[1] := n;
            t := n;
            h := 0;
11:         for l := 2 to k do
               r[l] := 0;
15:         if r[k] = n then
               mtc := false
            else
               mtc := true;
            goto 31;
20:         if t > 1 then
               begin
                  h := 0;
               end;
30:         h := h + 1;
            t := r[h];
            r[h] := 0;
            r[1] := t - 1;
            r[h+1] := r[h+1] + 1;
            goto 15;
         end;
31:      l := 1;
    end;


{*******************************************************}
{                                                       }
{  GET_COMBINATIONS                                     }
{                                                       }
{  This procedure controls the execution of the        }
{  procedure COMB and generates the values w, x, y,    }
{  and z.  These variables correspond to early-late,   }
{  early-early, late-early, and late-late packets      }
{  respectively.  All combinations given a total       }
{  number of interferers n are placed into the matrix  }
{  vector.                                              }
{                                                       }
{*******************************************************}

procedure GET_COMBINATIONS ( var vector:comb_matrix;
                             n:integer; var vector_length:
                             integer);

   var
      i,
      j,
      w,
      x,
      y,
      z:integer;

      mtc:boolean;

      v:index;

   begin
      for i := 1 to 4 do
         v[i] := 0;

      i := 0;
      mtc := true;

      if archive then
         writeln (outfilevar,'COMBINATIONS
       *******************');
      while mtc do
         begin
            if i = 0 then
               mtc := false;
            i := i + 1;
            comb (n,4,v,mtc);
            for j := 1 to 4 do
               vector[i,j] := v[j];
            w := v[1];
            x := v[2];
            y := v[3];
            z := v[4];
{            writeln (w,' ',x,' ',y,' ',z);}
            if archive then
               begin
                  writeln (outfilevar,w,' ',x,' ',y,' ',z);
               end;
         end;
      vector_length := i;
      if archive then
```

```
        writeln
(outfilevar,'*********************************');
   end;
{*******************************************************}
{                                                       }
{  MAIN                                                 }
{                                                       }
{  Program execution begins with a query for an         }
{  archive data file.  If the archive option is chosen }
{  all screen I/O will be echoed to the file given.    }
{  The next program query is for the maximum number of }
{  packets, n.  The program will then generate all     }
{  combinations of interferer packet types for a given }
{  value of n.  Each combination is then fed one by    }
{  one into the GENERATE_REALIZATION procedure where   }
{  all possible realizations are generated.  This list }
{  of realizations is then filtered by FILTER to cull  }
{  out invalid realizations.  A matrix of valid        }
{  realizations is fed to GET_Cj in order to generate  }
{  the Cj table.  This process is repeated for all     }
{  values of n between 1 and the maximum given by the  }
{  user.                                                }
{                                                       }
{*******************************************************}




begin
   archive := false;

   write ('DO YOU WANT A DATA ARCHIVE (y or n) -> ');
   readln (ans);
   if ans = 'y' then
      begin
         archive := true;
         write ('ENTER DATA FILE NAME -> ');
         readln (file_name);
         assign (outfilevar, file_name);
         rewrite (outfilevar);
      end;

   write ('ENTER THE MAX # OF PKTS PER REALIZATION -> ');
   readln (runs);

   initialize_cj_matrix;
   for n := 1 to runs do
      begin
```

```
          for i := 1 to 220 do
          for j := 1 to 4 do
             vector[i,j] := 0;

          if archive then
             begin
                writeln
(outfilevar,'***********************
                                 ***************************');
                writeln (outfilevar,'NUMBER OF PACKETS = ',n);
                writeln;
             end;

          get_combinations (vector,n,vector_length);
          for i := 1 to vector_length do
             begin
                init_interferers := vector[i,1]+vector[i,2];
                final_interferers := vector[i,1]+vector[i,4];
                max_interferers := vector[i,1] + vector[i,2] +
                               vector[i,3] + vector[i,4];
                min_interferers := vector[i,1];
                num_intervals := vector[i,2] + 2*vector[i,3] +
                             vector[i,4] + 1;

                if archive then
                   begin
writeln (outfilevar);
writeln (outfilevar);
writeln (outfilevar,'--------------------------');
writeln (outfilevar);
writeln (outfilevar,'PACKET CONFIGURATION IS ',vector[i,1],
                 ' ',vector[i,2],' ',vector[i,3],'
            ',vector[i,4]);
writeln (outfilevar);
writeln (outfilevar,'INITIAL PKTS = ',init_interferers);
writeln (outfilevar,'FINAL PKTS   = ',final_interferers);
writeln (outfilevar,'MAX # PKTS   = ',max_interferers);
writeln (outfilevar,'MIN # PKTS   = ',min_interferers);
writeln (outfilevar,'NUM INTERVALS= ',num_intervals);
writeln (outfilevar);
writeln (outfilevar,'--------------------------');
                   end;


                init_realization_matrix;

                generate_realizations (init_interferers,
                       max_interferers,
                                 num_intervals, min_interferers,
                                 active_matrix,
              active_matrix_length);
```

```
                filter (active_matrix_length, num_intervals,
                  final_interferers,
                        active_matrix);

                get_cj (active_matrix, active_matrix_length,
                  max_interferers);
             end;
       end;
    print_cj;
    close (outfilevar);
```

APPENDIX B

DERIVATION OF THROUGHPUT EQUATION FOR VARIABLE PACKET ALOHA

This appendix presents the derivation of the throughput equation for variable packet Aloha with power capture.

As seen in Chapter II equation (2.9), the probability of a tagged packet successfully being transmitted is

$$p_s(x,u) = e^{-g(x+u)}[1 + \sum_{n=1}^{\infty} f(n)2^{-(n+1)} \frac{[g(x+u)]^n}{n!} \sum_{j=0}^{N} C_j(n)] \tag{B.1}$$

Assign the following identity

$$h(n) = f(n)2^{-(n+1)} \frac{g^n}{n!} \sum_{j=0}^{N} C_j(n) \tag{B.2}$$

As described in Chapter II equations (2.10) and (2.11), the conditional channel departure rate and the channel throughput are given by the following:

$$\phi(x,u) = g(x+u)p_s(x,u) \tag{B.3}$$

$$S = \frac{1}{2} \int_0^{\infty} \int_0^{\infty} \phi(x,u)a(x)a(u)dxdu \tag{B.4}$$

Utilizing the binomial identity

$$(x+u)^n = \sum_{k=0}^{n} \binom{n}{k} x^k u^{n-k} \tag{B.5}$$

and expanding the exponential term in (B.1), the equation for $p_s(x,u)$ can be rewritten as

$$p_s(x,u) = \epsilon^{-gx}\epsilon^{-gu}\left[1 + \sum_{n=1}^{\infty} h(n) \sum_{k=0}^{n} \binom{n}{k} x^k u^{n-k}\right] \qquad \text{(B.6)}$$

Expansion of $\phi(x,u)$ in terms of $p_s(x,u)$ results in

$$\phi(x,u) = g(x+u)\epsilon^{-gx}\epsilon^{-gu}\left[1 + \sum_{n=1}^{\infty} h(n) \sum_{k=0}^{n} \binom{n}{k} x^k u^{n-k}\right] \qquad \text{(B.7)}$$

Multiplying produces

$$\begin{aligned} \phi(x,u) &= [gx\epsilon^{-gx}\epsilon^{-gu} + gu\epsilon^{-gu}\epsilon^{-gx}] \\ &\quad + [gx\epsilon^{-gu}\epsilon^{-gx} + gu\epsilon^{-gu}\epsilon^{-gx}] \cdot \left[\sum_{n=1}^{\infty} h(n) \sum_{k=0}^{n} \binom{n}{k} x^k u^{n-k}\right] \end{aligned} \qquad \text{(B.8)}$$

Using this definition of $\phi(x,u)$ in the equation for S (B.4) produces two separate double integrals. The first is equivalent to the equation (27) in Ref. 2. Assuming that $p_1(x) = 0$, $p_2(y) = 0$ and that the density function $a(x)$ is exponential, this first term for S is found in Ref. 2 to be

$$S_1 = \frac{G}{(G+1)^3} \qquad \text{(B.9)}$$

The remaining double integral remains to be evaluated. This term represents the increase in throughput due to power capture. The term is written here as

$$\begin{aligned} S_2 &= \frac{1}{2}\int_0^{\infty}\int_0^{\infty} [gx\epsilon^{-gx}\epsilon^{-gu} + gu\epsilon^{-gu}\epsilon^{-gx}] \\ &\quad \cdot \left[\sum_{n=1}^{\infty} h(n) \sum_{k=0}^{n} \binom{n}{k} x^k u^{n-k}\right] a(x)a(u)dxdu \end{aligned} \qquad \text{(B.10)}$$

Manipulating terms in the above equation produces

$$
\begin{aligned}
S_2 &= \frac{1}{2}\sum_{n=1}^{\infty} h(n)\left\{\int_0^{\infty}\int_0^{\infty} gx\epsilon^{-gx}\epsilon^{-gu}\sum_{k=0}^{n}\left[\binom{n}{k} x^k u^{n-k}\right] a(x)a(u)dxdu\right. \\
&\left. + \int_0^{\infty}\int_0^{\infty} gu\epsilon^{-gu}\epsilon^{-gx}\sum_{k=0}^{n}\left[\binom{n}{k} x^k u^{n-k}\right] a(x)a(u)dxdu\right\} \qquad \text{(B.11)}
\end{aligned}
$$

Further collection of like terms produces

$$
\begin{aligned}
S_2 &= \frac{1}{2}g\sum_{n=1}^{\infty} h(n)\left\{\sum_{k=0}^{n}\binom{n}{k}\int_0^{\infty} x^{k+1}\epsilon^{-gx}a(x)dx \cdot \int_0^{\infty} u^{n-k}\epsilon^{-gu}a(u)du\right. \\
&\left. + \sum_{k=0}^{n}\binom{n}{k}\int_0^{\infty} u^{n-k+1}\epsilon^{-gu}a(u)du \cdot \int_0^{\infty} x^k\epsilon^{-gx}a(x)dx\right\} \qquad \text{(B.12)}
\end{aligned}
$$

Each of the individual integrals above can be solved using the following identity [Ref. 5:p. 310].

$$\int_0^{\infty} x^n\epsilon^{-\mu x}dx = n!\mu^{-(n+1)} \qquad \text{(B.13)}$$

Additionally, the following substitutions can be made because packet size is assumed to be exponentially distributed and because the quantity G is given in packets/average packet length.

$$a(x) = \frac{1}{\overline{x}}\epsilon^{-x/\overline{x}} \qquad \text{(B.14)}$$

$$G = g\overline{x} \qquad \text{(B.15)}$$

Combining the integral identity in equation (B.13) with the definition of $a(x)$ produces this solution for the first integral term in equation (B.12).

$$\int_0^{\infty} x^{k+1}\epsilon^{-gx}a(x)dx = \frac{1}{\overline{x}}\int_0^{\infty} x^{k+1}\epsilon^{-(g+\frac{1}{\overline{x}})x}dx$$

$$\begin{aligned}
&= \frac{(k+1)!}{\overline{x}}(g+\frac{1}{\overline{x}})^{-(k+2)} \\
&= \frac{(k+1)!}{\overline{x}}(\frac{g\overline{x}+1}{\overline{x}})^{-(k+2)} \\
&= \frac{(k+1)!}{\overline{x}^{-(k+1)}}(G+1)^{-(k+2)} \qquad \text{(B.16)}
\end{aligned}$$

The second term in equation (B.12) is evaluated in the same manner as above.

$$\begin{aligned}
\int_0^\infty u^{n-k}e^{-gu}a(u)du &= \frac{1}{\overline{u}}\int_0^\infty u^{n-k}e^{-(g+\frac{1}{\overline{u}})u}du \\
&= \frac{(n-k)!}{\overline{u}}(g+\frac{1}{\overline{u}})^{-(n-k+1)} \\
&= \frac{(n-k)!}{\overline{u}}(\frac{g\overline{u}+1}{\overline{u}})^{-(n-k+1)} \qquad \text{(B.17)}
\end{aligned}$$

In this model, all packets come from the same distribution, therefore $\overline{x} = \overline{u}$. This modifies the above equation.

$$\int_0^\infty u^{n-k}e^{-gu}a(u)du = \frac{(n-k)!}{\overline{x}^{-(n-k)}}(G+1)^{-(n-k+1)} \qquad \text{(B.18)}$$

The third integral term in equation (B.12) is reduced as follows

$$\int_0^\infty u^{n-k+1}e^{-gu}a(u)du = \frac{(n-k+1)!}{\overline{x}^{-(n-k+1)}}(G+1)^{-(n-k+2)} \qquad \text{(B.19)}$$

The final integral term is as follows

$$\int_0^\infty x^{n}e^{-gx}a(x)dx = \frac{k!}{\overline{x}^{-k}}(G+1)^{-(k+1)} \qquad \text{(B.20)}$$

The total throughput of the system is

$$S = S_1 + S_2 \qquad \text{(B.21)}$$

Collecting equations (B.9), (B.11), and (B.16)-(B.20) produces

$$S = \frac{G}{(G+1)^3} + \frac{1}{2}g\sum_{n=1}^{\infty} h(n)\left\{\sum_{k=0}^{n} \frac{n!}{k!(n-k)!}\left[\frac{(k+1)!}{\overline{x}^{-(k+1)}}(G+1)^{-(k+2)}\right.\right.$$
$$\cdot\frac{(n-k)!}{\overline{x}^{-(n-k)}}(G+1)^{-(n-k+1)}$$
$$\left.\left. + \frac{(n-k+1)!}{\overline{x}^{-(n-k+1)}}(G+1)^{-(n-k+2)}\frac{k!}{\overline{x}^{-k}}(G+1)^{-(k+1)}\right]\right\} \qquad \text{(B.22)}$$

This simplifies to

$$S = \frac{G}{(G+1)^3} + \frac{1}{2}g\sum_{n=1}^{\infty} h(n)\left\{\sum_{k=0}^{n}\left[n!(k+1)\frac{(G+1)^{-(n+3)}}{\overline{x}^{-(n+1)}}\right.\right.$$
$$\left.\left. + n!(n-k+1)\frac{(G+1)^{-(n+3)}}{\overline{x}^{-(n+1)}}\right]\right\} \qquad \text{(B.23)}$$

Further simplification produces

$$S = \frac{G}{(G+1)^3} + \frac{1}{2}g\sum_{n=1}^{\infty} h(n)\frac{(G+1)^{-(n+3)}}{\overline{x}^{-(n+1)}}\sum_{k=0}^{n} n!(n+2) \qquad \text{(B.24)}$$

The final summation in the above equation is over the range of $k = 0$ to ∞. Because there are no k terms within the summation, the entire summation reduces to $n!n(n+2)$. Combining terms produces

$$S = \frac{G}{(G+1)^3} + \frac{1}{2}g\sum_{n=1}^{\infty}\frac{f(n)}{2^{n+1}}\sum_{j=0}^{N}[C_j(n)]\frac{g^n}{n!}\frac{(G+1)^{-(n+3)}}{\overline{x}^{-(n+1)}}n(n+2)n! \qquad \text{(B.25)}$$

Combining terms and utilizing the identity $G = g\overline{x}$ produces the following solution for the throughput of variable packet Aloha with power capture.

$$S = \frac{G}{(G+1)^3} + \sum_{n=1}^{\infty}\frac{n(n+2)}{2^{n+2}}f(n)G^{n+1}(G+1)^{-(n+3)}\sum_{j=0}^{N}C_j(n) \qquad \text{(B.26)}$$

APPENDIX C

DERIVATION OF THROUGHPUT EQUATION FOR SELECTIVE-REPEAT ALOHA

This appendix presents the derivation of the throughput equation for the variable packet Aloha network with power capture and selective-repeat data link control presented in Chapter III.

From Chapter III equation (3.2), the probability of tagged packet success to be

$$p_s(z) = e^{-g(mz+m)}\left[1 + \sum_{n=1}^{\infty} f(n)2^{-(n+1)}\frac{[g(mz+m)]^n}{n!}\sum_{j=0}^{N} C_j(n)\right] \qquad (C.27)$$

The conditional probability of successful minipacket transmission is given in equation (3.3) of Chapter III to be

$$q(n,y) = p_s(z)[1 - p_1(m)][1 - p_2(y)] \qquad (C.28)$$

Also given in Chapter III is the conditional minipacket departure rate in the interval of $n+1$ minipackets. This equation is rewritten here.

$$\rho_m(z,y) = mg(z+1)q(z,y) \qquad (C.29)$$

Equation (3.5) in Chapter III gives the expression for the average minipacket throughput and is rewritten here.

$$S_m = \frac{\bar{z}}{\bar{z}+1}\int_0^{\infty} \rho_m(z,y)dy \qquad (C.30)$$

The equation for minipacket throughput given above is now expanded to produce the following:

$$\begin{aligned}
S_m &= \frac{\bar{z}}{\bar{z}+1} mg[1-p_1(m)]\epsilon^{-mg} \int_0^\infty [1-p_2(y)]b(y)dy \cdot \sum_{z=1}^{\infty}(z+1)D(z)\epsilon^{-mzg} \\
&+ \frac{\bar{z}}{\bar{z}+1} mg[1-p_1(m)]\epsilon^{-mg} \int_0^\infty [1-p_2(y)]b(y)dy \cdot \sum_{z=1}^{\infty} \cdot \sum_{n=1}^{\infty} f(n)2^{-(n+1)} \\
&\cdot \frac{[g(mz+m)]^n}{n!} \sum_{j=0}^{N} [C_j(n)]\,(z+1)D(z)\epsilon^{-mzg} \qquad \text{(C.31)}
\end{aligned}$$

where $D(z)$ is the probability that a packet consists of z minipackets.

The first term in the above equation is the same as the equation evaluated in Ref. 2 equation (36). The resulting throughput equation for this term is given in Ref. 2 equation (37b). Call this part of the total throughput S_1. Then from Ref. 2.

$$S_1 = \frac{\sigma G v^2(2-v)[1-p_1(m)]}{\bar{z}(\bar{z}+1)w^2(1-v)^2} \qquad \text{(C.32)}$$

With the following definitions:

$$\begin{aligned}
w &= 1 - \frac{1}{\bar{z}} && \text{(C.33)} \\
v &= w\epsilon^{-\frac{G}{\bar{z}}} && \text{(C.34)} \\
G &= m\bar{z}g && \text{(C.35)} \\
\sigma &= \int_0^\infty [1-p_2(y)]b(y)dy && \text{(C.36)}
\end{aligned}$$

Let S_2 be the remaining term of the total throughput given in equation (C.31). By separating terms and noticing that $[g(mz+m)]^n = (gm)^n(z+1)^n$, S_2 is rewritten as

$$S_2 = \frac{m\bar{z}g}{\bar{z}+1}[1-p_1(m)]\epsilon^{-mg}\sigma$$

$$\cdot \sum_{n=1}^{\infty}\left[f(n)2^{-(n+1)}\frac{(gm)^n}{n!}\sum_{j=0}^{N}[C_j(n)]\cdot\sum_{z=1}^{\infty}(z+1)^{n+1}D(z)\epsilon^{-mzg}\right] \quad (C.37)$$

As stated in Chapter III equation (3.7), assume that $D(z)$ is geometrically distributed and is defined as follows:

$$D(z) = \frac{1}{\bar{z}}(1-\frac{1}{\bar{z}})^{z-1} \quad (C.38)$$

Utilizing the equations for G, $D(z)$, and w the equation for S_2 can be written as follows:

$$\begin{aligned} S_2 &= \frac{G}{\bar{z}+1}[1-p_1(m)]\epsilon^{-(\frac{G}{\bar{z}})}\sigma \\ &\cdot \sum_{n=1}^{\infty} f(n)2^{-(n+1)}\frac{(\frac{G}{\bar{z}})^n}{n!}\sum_{j=0}^{N}[C_j(n)]\cdot\sum_{z=1}^{\infty}(z+1)^{n+1}w^{z-1}\epsilon^{-z(\frac{G}{\bar{z}})} \quad (C.39)\end{aligned}$$

Assign the following identity:

$$h(n) = f(n)2^{-(n+1)}\frac{G^n}{n!}\frac{1}{\bar{z}^{n+1}}\sum_{j=0}^{N}C_j(n) \quad (C.40)$$

At this point, consolidate the exponential terms and divide out two of the w terms to equalize the exponents within the summation over z.

$$S_2 = \frac{G}{\bar{z}+1}[1-p_1(m)]\sigma\sum_{n=1}^{\infty}h(n)w^{-2}\sum_{z=0}^{\infty}(z+1)^{n+1}w^{z+1}\epsilon^{-(\frac{G}{\bar{z}})(z+1)} \quad (C.41)$$

Assign the following identities:

$$k = (z+1) \quad (C.42)$$

$$t = (n+1) \quad (C.43)$$

This produces the following

$$S_2 = \frac{G}{\overline{z}+1}[1 - p_1(m)]\sigma \sum_{n=1}^{\infty} h(n) w^{-2} \sum_{k=2}^{\infty} k^t v^k \tag{C.44}$$

At this point, modify the summation over k such that the range is from $k = 1$ to ∞. To do this, subtract off from the total of the summation the value of the summation for $k = 1$ which is v in this case. Therefore, the above equation is changed to

$$S_2 = \frac{G}{\overline{z}+1}[1 - p_1(m)]\sigma \sum_{n=1}^{\infty} h(n) w^{-2} \left[\sum_{k=1}^{\infty} \left[k^t v^k\right] - v\right] \tag{C.45}$$

Given that $|v| \leq 1$, use the following mathematical identity to solve the above equation [Ref. 6: p. 142].

$$\sum_{k=1}^{\infty} k^t v^k = (1-v)^{-(t+1)}(t+1)! \sum_{r=1}^{t} v^r \sum_{k=0}^{r-1} (-1)^k \frac{(r-t)^t}{k!(t-k+1)!} \tag{C.46}$$

Using the above identity, the final form of S_2 is given as

$$\begin{aligned} S_2 \;=\; & \frac{G}{\overline{z}+1}[1 - p_1(m)]\sigma \sum_{n=1}^{\infty} h(n) w^{-2} \\ & \cdot \left\{\left[(1-v)^{-(t+1)}(t+1)! \sum_{r=1}^{t} v^r \sum_{k=0}^{r-1} (-1)^k \frac{(r-k)^t}{k!(t-k+1)!}\right] - v\right\} \end{aligned} \tag{C.47}$$

The total throughput for selective-repeat Aloha with power capture is given as

$$S = S_1 + S_2 \tag{C.48}$$

APPENDIX D

ANALYTICAL AND NUMERICAL RESULTS FOR NEAR/FAR EFFECT CAPTURE

This appendix provides the details of the analytical and numerical methods used to determine the values for $Pr\{capture|n\}$ in Chapter III. First, the derivation of $Pr\{capture|n = 1\}$ is provided. This is followed by a MATHCAD file which was used to calculate capture for $n = 2$ and 3.

A. ANALYTICAL METHOD

The case of $n = 1$ is sufficiently simple to conduct a straightforward mathematical derivation of the capture probability given the near/far effect. From Chapter IV the following is known:

$$Z = \frac{X}{Y} \tag{D.49}$$

$$f_X(x) = \frac{2\pi}{\alpha G} x^{-(2/\alpha+1)} \tag{D.50}$$

$$f_Y(y|n) = \left[\frac{2\pi}{\alpha G} y^{-(2/\alpha+1)}\right]^{\circledast n} \tag{D.51}$$

For the case of $n = 1$, both y and x are defined between the values of $r_{max}^{-\alpha}$ and ∞. Also from Chapter IV, equation (4.10), the definition of $f_Z(z|n)$. From this equation, it is clear that both yz and y must be greater than $r_{max}^{-\alpha}$ given the valid ranges for $f_X(x)$ and $f_Y(y|n)$. This implies that for values of z greater than 1, the valid range for y is $y \geq r_{max}^{-\alpha}$. Therefore, the equation for $f_Z(z|n)$ with the proper limits of integration is

$$f_Z(z|n=1) = \int_{r_{max}^{-\alpha}}^{\infty} y f_X(yz) f_Y(y|n=1)dy \tag{D.52}$$

$$= \int_{r_{max}^{-\alpha}}^{\infty} y \frac{2\pi}{\alpha G}(yz)^{-(2/\alpha+1)} \frac{2\pi}{\alpha G} y^{-(2/\alpha+1)} dy \tag{D.53}$$

$$\left(\frac{2\pi}{\alpha G}\right)^2 z^{-(2/\alpha+1)} \int_{r_{max}^{-\alpha}}^{\infty} y^{-(4/\alpha+1)} dy \tag{D.54}$$

$$= \left(\frac{2\pi}{\alpha G}\right)^2 z^{-(2/\alpha+1)} \left(-\frac{\alpha y^{-4/\alpha}}{4}\right)\Bigg|_{r_{max}^{-\alpha}}^{\infty} \tag{D.55}$$

Remembering that $G = \pi r_{max}^2$ gives us

$$f_Z(z|n=1) = -\frac{1}{\alpha r_{max}^4} z^{-(2/\alpha+1)} \, y^{-4/\alpha}\Big|_{r_{max}^{-\alpha}}^{\infty} \tag{D.56}$$

$$= \frac{1}{\alpha} z^{-(2/\alpha+1)} \tag{D.57}$$

To determine the capture probability from the above distribution, simply integrate from the capture threshold γ_0 to *infinity*.

$$Pr\{capture|n=1\} = Pr\{z \geq \gamma_0 | n=1\} \tag{D.58}$$

$$= \int_{\gamma_0}^{\infty} \frac{1}{\alpha} z^{-(2/\alpha+1)} dz \tag{D.59}$$

$$= \frac{1}{2} \gamma_0^{-2/\alpha} \tag{D.60}$$

B. NUMERICAL METHODS

This document is a MATHCAD file used to numerically calculate the values of P(capture|n) for values of n from 2 to 3. The general approach is to use a numerical convolution to determine the n'th convolution of f(y). These convolution results are then used in numerical integrations to generate the P(capture|n) results.

Initialize and Define Variables

$M \equiv 1$ (M corresponds to r_{max})

$\alpha \equiv 4$

$G \equiv \pi \cdot M^2$

$D \equiv 4$ (2 < D < 32 sets the number of convolution points.)

$i \equiv 0 \,..32 \cdot D - 1$

$$v_i \equiv \frac{2 \cdot i + 1}{2 \cdot D}$$

$$f(t) \equiv 2 \cdot \frac{\pi}{\alpha \cdot G} \cdot t^{-\left[\frac{2}{\alpha}+1\right]} \cdot \left[\Phi(10 - t) - \Phi\left[M^{-\alpha} - t\right]\right]$$

$h(t) \equiv f(t)$

f(t) corresponds to f(w) in the model

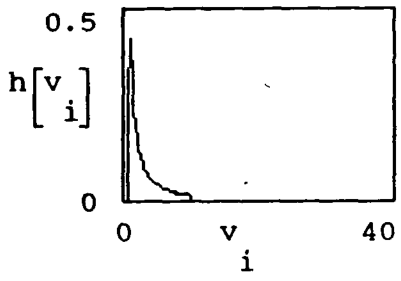
$p(t) \equiv f(t)$

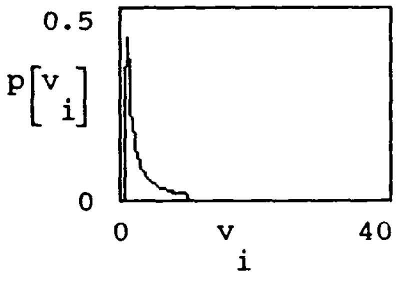
The following equation performs a discrete convolution on the functions defined as X and Y.

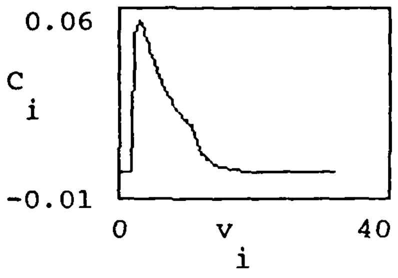
$$\mathrm{conv}(X,Y) \equiv \left[\frac{1}{D}\right] \cdot \mathrm{icfft}\left[\sqrt{16 \cdot D} \cdot \overrightarrow{(\mathrm{cfft}(X) \cdot \mathrm{cfft}(Y))}\right]$$

Step 1. Convolve f(t) with itself to get f(y|n=2) (call this C) and plot the resulting functions.

$$C \equiv \text{conv}\left[\overrightarrow{h(v)}, \overrightarrow{p(v)}\right]$$

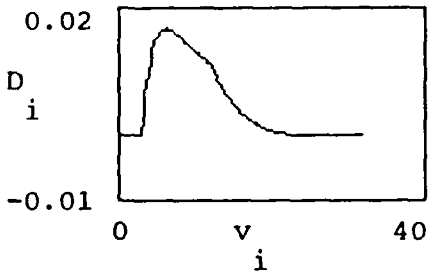






Step 2. Convolve f(y|n=2) with f(t) to get f(y|n=3) (called D) and plot the relulting function.

$$D \equiv \text{conv}\left[\overrightarrow{h(v)}, \vec{C}\right]$$



Step 3. Continue in the above manner for as many cases of f(y|n) as needed.

Step 4. Calculate f(z|n) using a linear interpolation of the numerical value for f(y|n).

$$f(z) := 2\cdot\frac{\pi}{\alpha\cdot G}\cdot z^{-\left[\frac{2}{\alpha}+1\right]}\cdot\int_{2\cdot M}^{30} y^{-\alpha}\,{}^{-\left[\frac{2}{\alpha}\right]}\cdot\mathrm{linterp}(v,C,y)\,dy$$

Step 5. Calculate the capture probability given n known interferers by integrating f(z|n) (f(z) in this example) from the capture threshold (3 in this example) to a maximum usefull limit which is determined by the range of values for which f(y|n) is valid. In this case, 30 is the upper limit.

$$Z := \int_{3}^{30} f(z)\,dz$$

$Z = 0.057$

Step 6. Continue on in the same manner as above to calculate the capture probability for different values of n. The following equation defines f(z|n) for the case where n=3.

$$g(h) := 2\cdot\frac{\pi}{\alpha\cdot G}\cdot h^{-\left[\frac{2}{\alpha}+1\right]}\cdot\int_{3\cdot M}^{30} y^{-\alpha}\,{}^{-\left[\frac{2}{\alpha}\right]}\cdot\mathrm{linterp}(v,D,y)\,dy$$

$$H := \int_{3}^{30} g(h)\,dh$$ This equation calculates P(capture|n=3)

$H = 0.022$

REFERENCES

[1] Tannenbaum, Andrew S., *Computer Networks,* pp. 253-257, Prentice Hall Inc, 1981.

[2] Arnbak, Jens C., and van Blitterswijk, Wim, "Capacity of Slotted Aloha in Reyleigh Fading Channels," *IEEE Journal on Selected Areas in Communications,* Vol. SAC-5, No. 2, pp. 261- 269, 1987.

[3] Ha, Tri T., "Aloha VSAT Networks with Data Link Control," *ICC '88,* Vol. 25, No. 3, pp. 1-4, 1988.

[4] Borchardt, Randy L., *Performance Analysis of Aloha Networks Utilizing Multiple Signal Power Levels,* Master's Thesis, Naval Postgraduate School, Monterey, California, June 1988.

[5] Gradshteyn, I. S., and Ryzhik, I. M., *Table of Integrals, Series, and Products,* pp. 310, Academic Press, 1980.

[6] Hanson, Eldon, *A Table of Series and Products,* pp. 142, Prentice Hall Inc, 1975.

[7] Nijenhuis, Albert, and Wilf, Herbert S., *Combinatorial Algorithms,* pp. 46-51, Academic Press, 1975.

INITIAL DISTRIBUTION LIST

		No. Copies
1	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3	Director for Command. Control and Communications Systems, Joint Staff Washington, DC 20318-6000	1
4	Superintendent, Code 74 C3 Academic Group Naval Postgraduate School Monterey, California 93943-5000	1
5	AFIT/NR Wright-Patterson AFB, Ohio 45433-6583	1
6	AFIT/CIRK Wright-Patterson AFB, Ohio 45433-6583	1
7	Professor Tri T. Ha, Code 62Ha Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	5
8	Professor Glen A. Myers, Code 62Mv Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
9	Captain Joseph T. McCartin HQ USEUCOM/J6 APO, New York 09128	2